SOME HEURISTIC APPROACHES FOR MINIMIZING THE TOTAL NUMBER OF TARDY JOBS FOR MXN FLOW SHOP SCHEDULING PROBLEM

By

JITENDRA V. DESAI

INDUSTRIAL AND MANAGEMENT ENGINEEING PROGRAMME

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

AUGUST, 1980

SOME HEURISTIC APPROACHES FOR MINIMIZING THE TOTAL NUMBER OF TARDY JOBS FOR MXN FLOW SHOP SCHEDULING PROBLEM

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY

By
JITENDRA V. DESAI

to the
INDUSTRIAL AND MANAGEMENT ENGINEEING PROGRAMME
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

AUGUST, 1980

IMEP-1980-M-DES-SOM

CERTIFICATE

This is to certify that the present work on "Some heuristic approaches for mxn flow shop scheduling problem for minimizing the total number of tardy jobs" by Jitendra V. Desai has been carried out under my supervision and has not been submitted elsewhere for the award of a degree.

August, 1980

(J.L. Batra)
Professor & Head
Industrial&Management Engineering
I.I.T. Kanpur.

ACKNOWLEDGEMENTS

It is indeed a great pleasure to express my deep sense of gratitude to Dr. J.L. Batra for his constant encouragement, guidance and constructive criticisms, rendered throughout the course of the work. His readiness to devote his valuable time at any odd hour despite being very busy, has made this work attain its present stature.

I also wish to express my indebtedness to Dr. S. Sadagopan and Dr. (Mrs) Prabha Sharma who had been a constant source of inspiration and encouragement throughout my stay at IIT Kanpur.

With great pleasure, I thank all my friends whose pleasant company made my stay at IIT, Kanpur, a joyful and an unforgettable one.

I am also thankful to Mr. C.M. Abraham for his neat and accurate typing and Mr. Buddhiram Kandiyal for his immaculate cyclostyling work.

JITENDRA V. DESAI

CONTENTS

SYNOPSIS

The present work deals with m-machine n-job flow shop scheduling problem considering the total number of tardy jobs as the effectiveness criterion. Four heuristic algorithms are proposed and have been tested on 60 randomly generated problems of 12 different sizes. The performances of the heuristics have also been tested for a special class of ordered flow shop problem . All the four heuristics gave the same results for the above special class of the ordered flow shop problem, considered in the present work. The performances of the heuristics were evaluated based on average number of ~~tardy~~ tardy jobs, number of times best solution was generated, the frequency of obtaining guaranteed optimal solution and the average execution time for various problem sizes. The analysis indicated that, heuristic-4 gives overall best performance. Further, it was observed that the number of jobs have more significant effect on the computational times than the number of machines. In case of all the heuristics the average execution time is very small. It is suggested that a problem should be solved using all the four heuristics and the best solution should be adapted since no heuristic explicitly guarantte the best solution.

CHAPTER 1

INTRODUCTION

Scheduling can be defined as the allocation of resources over time to perform a collection of tasks. In a typical production situation, where the resources are generally machines and the tasks are jobs, scheduling decisions are generally reached by using the scientific and systematic approaches involving the following four steps.

1) Formulation
2) Analysis
3) Synthesis
4) Evaluation

In the first stage, basically a problem is identified and the criteria that should guide decision-making are determined. This is often a subtle and complicated activity, but good decisions can seldom be expected without a clear definition of the problem at hand and an explicit recognition of objectives. Analysis is the detailed process of examining the elements of a problem and their interrelationships. This stage is aimed at identifying the decision variables and also at specifying the relationships among them and the constraints they must obey. Synthesis is the process of building alternative solutions to the problem. Its role is to characterize the feasible options that are

available. Finally, evaluation is the process of comparing these feasible alternatives and selecting a desirable course of action. This selection, is, of course, based on the criteria that are developed at the outset.

Ideally, the objective function should consist of all the system costs that depend on scheduling decisions. In practice, however, such costs are difficult to measure, identify and incorporated completely. Moreover, objective functions consisting of more than single type of costs (which may be contradictory in nature) sometimes become too complex to be analysed. Broadly, three types of decision making goals seem to be prevalent in scheduling : (i) efficient utilization of resources, (ii) rapid response to demands and (iii) close conformance to prescribed dead lines. Frequently an important cost related measure of system performance (such as, machine idle time, job waiting time, job lateness, etc.) can be used as a substitute for the total system cost, and quantitative approaches to problems with these criteria appear throughout the literature on scheduling.

Two kinds of constraints are generally encountered in scheduling problems. First, there are limits on the capacity of available resources, and, second there are technological restrictions on the order in which tasks can be performed.

Any solution to scheduling problem has to answer the

two major questions.

1) Which resources will be allocated to perform each task?

2) When will each task be performed?

In other words, the essence of scheduling problem gives rise to allocation and sequencing decisions. The present work deals with the multistage flow shop scheduling problem. In flow shop jobs are multistage in nature (a job is a set of operations with the special precedence relations) and machines have serial configuration, i.e., it is possible to number the machines so that if the ith operation of any job j precedes its kth operation then the machine required by ith operation has a lower number then the machine required by the kth operation. In particular each operation after the first has only exactly one direct predecessor and each operation before the last has exactly one successor. In general any flow shop scheduling problem can be defined as follows :

Given a flow shop environment with n jobs and m machines the objective is to determine an optimal sequence considering some measure of effectiveness. The effectiveness of any sequence can be measured in terms of the criterion of effectiveness. Some of the important effectiveness criteria are : make-span time, average completion time, due date performance, machine utilization, inventory of jobs in process etc. The due date performance of any sequence can be measured in terms of lateness of the job, total or mean tardiness,

maximum tardiness and total number of tardy jobs. Lateness of a job can be defined as the difference between its completion time and its due date. The maximum of lateness and zero is called tardiness. A job is called tardy or late if its tardiness is non-zero (i.e., strictly positive) and non-tardy or early otherwise.

Considerable effort has been directed by many researchers towards the solution of flow shop scheduling problem, subject to standard set of assumptions listed by Bakshi et al [1]. Most of the flow shop problems have been tackled using combinatorial optimization techniques, Monte Carlo sampling, random sampling, biased sampling, decomposition, heuristic approaches and complete enumeration techniques. Most commonly used combinatorial method is the Branch and Bound methodology. Minimization of the make span time and average flow time are the criteria considered most often. Jhonson [2] has suggested an algorithm which yields optimal solution for two machine n job flow shop case considering minimization of the make-span time. Number of heuristic approaches, have been reported in the literature, for the general mxn flow shop scheduling problem considering minimization of make span time as the effectiveness criterion. Some of the notable contributions are due to Campbell Dudek and Smith [3], Palmer [4], Gupta, J.N.D [5], Petrov [6], Nobesimha [7], Krone et al [8], Page [9] etc. Other criteria like minimizing the total or mean tardiness

minimizing the inventory of jobs in process, maximizing the machine utilization etc. have also been considered. Survey of the literature suggested that very little work has been done for general flow shop scheduling problems considering due date related criteria. In particular no algorithm has been reported for the minimization of the total number of tardy jobs. May be, this is due to the fact that the performance of algorithms considering the due date related criteria is not linearly related to the completion time as the value of the objective function depends upon two independent entities, viz., the completion time of the job (which is sequence dependent) and its due date (which is independent of the job sequence). Hence, in general, the analysis of the due date related criteria is more difficult as compared to other criteria.

Maxwell [10] has given an integer programming formulation for one machine n job flow shop problem to minimize the total number of tardy jobs. This formulation was questioned by Sidney [11] and he gave a counter example to show that the cuts (cutting planes) generated by Maxwell's procedure eliminate the optimal solution. Maxwell [12] has acknowledged that this algorithm does not generate the efficient cuts. For the counter example given by Sidney, he has shown that his procedure leads to an alternate optimal solution. Further,

Moore [13] claims that the proof given by Maxwell is incomplete and the number of variables and constraints expand rapidly and hence for the case of multistage flow shop problem this approach seems infeasible.

Moore [14] has given a simpler method for 1xn flow shop problem for the criterion of minimization of total number of tardy jobs. Many investigators [15], [16], [11], [14] have given proof of optimality of Moore's algorithm or its closely related form.

In the present work four heuristic approaches have been developed for the general mxn flow shop scheduling problem considering the criterion of minimization of the total number of tardy jobs. The conceptual frame work suggested by Moore for single machine case has been utilized effectively for the development of these heuristics. The proposed heuristics have also been tested for a special class of mxn flow shop scheduling problems, known as the problem with ordered processing time matrices.

The thesis is organised in three chapters. In Chapter II problem statement and solution methodology are presented. Computational experience with large number of randomly generated problems of machine size varying from 2 to 5 and the job size varying from 10 to 50 is presented in Chapter III alongwith the conclusions.

CHAPTER 2

PROBLEM FORMULATION AND SOLUTION METHODOLOGY

2.1 STATEMENT OF THE PROBLEM

Consider a multi-machine flow shop situation with m machines and n jobs. Each job is made up of a set of tasks or operations which are to be processed in a given technological order. All the standard assumptions given in [1] for a general flow shop environment are considered as valid and the objective is to minimize the total number of tardy jobs (N_T).

2.2 NOMENCLATURE

m	-	Number of machines
n	-	Number of jobs
(j)	-	jth position in any sequence
$t(i,j)$	-	Processing time of job (j) on machine i
$d(j)$	-	Due date of job (j)
$c(i,j)$	-	Completion time of job (j) on machine i
N_T	-	Total number of tardy jobs for the given sequence
$\{E\}$	-	Set of early or non-tardy jobs
$\{L\}$	-	Set of late or tardy jobs
$L(j)$	-	Lateness of job(j); $L(j) = c(m,j) - d(j)$
$T(j)$	-	Tardiness or positive lateness of job (j); $T(j) = \max\ \{0, L(j)\}$

IT(j) - Difference of the completion times of the job (j) and job (j-1) on the last machine; IT (j) = C(i,j) - C(i,j-1)

S(j) - Sum of the processing times of job (j) on all m machines;

$$S(j) = \sum_{i=1}^{m} t(i,j)$$

2.3 MATHEMATICAL FORMULATION

Mathematically, the problem can be stated as follows :

$$\text{Minimize } N_T = \sum_{j=1}^{n} F(T_*(j))$$

where $F(x) = 1$, if $x > 0$

$= 0$ otherwise.

Subject to the set of standard flow shop constraints as listed by Bakshi et al [1].

2.4 FOUNDATIONS OF PROPOSED METHODOLOGY

For the case of single machine, n job flow shop scheduling problem, Moore [14] proposed an algorithm for the minimization of the total number of tardy jobs. The algorithm alongwith the proof is summarized below. Moore's algorithm is based on the following result, the proof of which is self evident.

In any situation, if the EDD (early due date) sequence yields zero or exactly one tardy job, then it is the optimal sequence for the number of tardy jobs criterion. However, if it yields more than one tardy job the EDD sequence may not be optimal.

The algorithm suggested by Moore [14] assumes the form of the optimal sequence as follows :

a) First, a set {E* } of early or non-tardy jobs in EDD order

b) Then, a set {L* } of late or tardy jobs in any order.

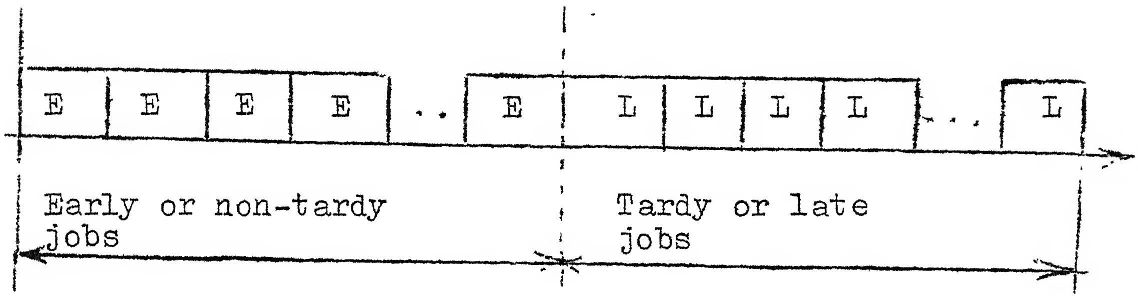


Figure 1 The form of a sequence that minimizes N_T

2.5 MOORE'S ALGORITHM

The various steps of the algorithm are :

Step-1 : Place all the jobs in set E according to EDD order. Let set L be empty.

Step-2 : If no jobs in set E are late, stop: E must be optimal, otherwise identify the first late job in set E. Suppose this turns out to be job (j).

Step-3 : Identify the longest job among the first j jobs in sequence. Remove this job from set E and place it in set L. Revise the completion times of the jobs remaining in set E and return to step-2.

2.5.1 Proof :

As the job (j-1) is non-tardy, its due date is greater than or equal to its completion time and as the job (j) is tardy, its due date is less than its completion time. Hence,

$$C(j-1) < d(j-1) \qquad (1)$$

and

$$C(j) > d(j) \qquad (2)$$

but

$$C(j-1) = \sum_{i=1}^{j-1} t(i) \qquad (3)$$

and

$$C(j) = \sum_{i=1}^{i=1} t(i) \qquad (4)$$

Substituting (3) and (4) in (1) and (2), we get,

$$\sum_{i=1}^{j} t(i) > d(j)$$

or

$$\sum_{i=1}^{j-1} t(i) + t(j) > dj \qquad (5)$$

and

$$\sum_{i=1}^{j-1} t(i) \leq d(j-1) \qquad (6)$$

After the job with the maximum processing time among the first j jobs is moved to set L, the completion time of job (j) would be reduced atleast by the amount t(j). Otherwise, if $\max\limits_{1 \leq i \leq j} t(i)$ occurs for i = j, then job (j) would be moved to set L. In the latter case the number of tardy jobs does not

increase as the job (j) was tardy anyway. In the former case, also the number of tardy jobs does not increase. Since in this case the completion time of job (j) is reduced by

$t(k) = \max_{1 \leq i \leq j} \{t(i)\}$ As t(k) is greater than t(j), its new completion time becomes less than the left hand side of eqn. (6), which is less than or equal to d(j-1) and hence job (j) becomes non-tardy.

2.6 PROPOSED METHODOLOGY

2.6.1 Development

As stated earlier in Chapter 1 the IP approach of Maxwell [10] has two main drawbacks even for 1xn problem. Firstly, the cuts (cutting planes) generated are very inefficient and therefore the greater computational effort is required. Secondly, the number of variables and constraints becomes prohibitively large as n increases. Thus, as such the IP approach has hardly any use even for the moderate size real life mxn flow shop problems. Hence, the only feasible alternative left is to go for a heuristic approach procedure which may yield fairly good solutions without requiring large computational effort.

Moore's algorithm [14] for 1xn problem is fairly simple and does not require large computational effort. As pointed out in Section 2.5, the algorithm requires the lenghiest job to be identified. The purpose of identifying the lengthiest job is that it would be removed from E and would be placed in set L

in order to reduce the completion time of the first tardy job to an extent that it becomes non-tardy. For a single machine case, it is assured that by shifting the lengthiest job to set L, atleast the first tardy job would become non-tardy as the lengthiest job is uniquely defined due to the absence of idle time. For a multimachine case, all the machines but for the first machine will generally have inserted idle times. The completion time of any job is made up not only of the processing time, but the idle time is also included in it. Hence, the job to be shifted to set L from among first j jobs where job (j) is the first tardy job should be the one which brings in maximum reduction in the completion time of the first tardy job. Let us term this job as the 'shift' job. But the main difficulty here is that the idle times are sequence dependent and would be changed if the sequence is changed. As such, one can not guarantee that by shifting any particular job to set L the first tardy job would become non-tardy. Therefore, the strategy to choose the shift job to be shifted to set L is very important. The value of the objective function is calculated after this job is shifted to set L and compared with the objective function value before the change in the sequence was made. The sequence giving the better value of the objective is retained.

Thus, the important differences between Moore's [14] single machine algorithm and the proposed heuristics are : (1) the value of the objective function is to be calculated in the proposed

methodology before and after the shift job is shifted to set L and only if the proposed change in the sequence brings in some improvement in the objective function value, it is to be carried out. Another important difference is the way in which the job to be shifted to set L is chosen. Four heuristic rules are developed for selecting the shift job, which form the basis of the proposed heuristics.

2.6.2 Heuristic - 1 :

The principle aim in selecting the shift job is to see that the removal of the shift job from E brings in the maximum attainable reduction in the completion time of the first tardy job and makes it non-tardy. Suppose job (j) is the first tardy job. Let

$$S(k) = \sum_{i=1}^{m} t(i,k)$$ denote the sum of the processing ti[me]

on all m machines for any job (k). Select the shift job as the one having the maximum value of $S(k)$ for $k = 1, 2, \ldots, j$. It is hoped that by choosing the job having maximum $S(k)$ as the shift job and removing it from E would make : 1) the job (j) non-tardy and 2) its new completion time not greater than the previous completion time of job (j-1). However, it is not assured that the shifting operation i.e., removal of the shift job from E would always accomplish objectives (1) and (2). Therefore, N_T is calculated before and after the shifting operation, If the proposed shifting operation increases N_T, job (j) becomes

the new shift job instead of job (k) and the shifting operation is carried out. If, however, objective (1) is accomplished and N_T remains the same by the proposed shifting, then the accomplishment of objective (2) is checked. If objective (2) is also accomplished then the proposed shifting operation is performed; otherwise job (j) becomes the shift job instead of job (k) and the shifting operation is carried out. However, if objective (1) is accomplished and N_T decreases by the proposed shifting, job (k) stays as the shift job and the shifting operation is carried out.

The methodology mentioned above for selecting the shift job is the basis of the present heuristic. All other steps are similar to Moore's [14] single machine algorithm.

The various steps of the algorithm are :

Step-1 : Place all the jobs in E according to EDD order. Let set L be empty.

Step-2 : If no jobs in E are late, stop; otherwise, identify the first tardy job in E. Let job (j) be the first tardy job. Go to step-3.

Step-3 : Identify the job having the maximum value of S(i) for i = 1,2, ... j. Suppose this is job (k) Go to step 4.

Step-4 : If j = k, remove this job from E, place it in L and go to step-5. Othersie, go to step-6.

Step-5 : Revise the completion times of all the jobs in E. Go to step-2.

Step-6 : Calculate N_T. Call it N_{T1}. Calculate N_T by removing job (k) from E and placing it in L. Call it N_{T2}.

There are three cases to be considered.

Case (i) : $N_{T1} > N_{T2}$

Remove job (k) from E and place it in L and go to step-5.

Case (ii): $N_{T1} < N_{T2}$.

Remove job (j) from E and place it in L and go to step-5.

Case (iii): $N_{T1} = N_{T2}$

Calculate C(m,j-1). Call it $C(m,j-1)_1$. Calculate C(m,j-1) by removing job (k) from E and placing it in L. Call it $C(m,j-1)_2$. If $C(m,j-1)_1 \geq C(m,j-1)_2$, remove job (k) from E and place it in L. Go to step-5. Otherwise remove job (j) from E and place it in L. Go to step-5.

2.6.3 Heuristic-2 :

Let us examine the general expression for the completion time of any job (i), which is given below :

$$C(m,i) = t(m,i) + \max\{C(m,i-1),\ C(m-1,\ i)\}$$

From the above expression it can be clearly seen that the term t(m,i) would always contribute to the completion time of job (i). Larger value of t(m,i) would result in larger C(m,i)

and subsequent delay in the following jobs. Therefore, if the job (k) having the maximum value of $t(m,i)$ for $i = 1, 2, \ldots j$ is selected as the shift job, it can be hoped that the shifting operation would accomplish objectives (1) and (2) stated in Section 2.6.2. As pointed out earlier, the proposed shifting is carried out only if it accomplishes objectives (1) and (2). Otherwise job (j), the first tardy job becomes the shift job instead of job (k) and the shifting operation is performed. This methodology of selecting the shift job is the basis of the present heuristic. The various steps of this algorithm are identica that of to heuristic-1 except step-3. Step-3 of the heuristic algorithm 2 is given below:

Step 3 : Identify the job, having the maximum value of $t(m,i)$ for $i = 1, 2, \ldots j$. Suppose this is job (k). Go to step 4.

2.6.4 Heuristic - 3 :

For any sequence the processing times of the jobs on the first machine play an important role to determine the completion times of the jobs. If a job has a larger processing time on the first machine, the subsequent machines have to be idle, waiting for the job to arrive from the first machine. This waiting time, in turn increases the job completion time and all the subsequent jobs are delayed. Hence, in order to reduce the waiting times and in turn completion times of all the subsequent

jobs in the sequence, the shift job should be the one having the maximum value of $t(1,i)$, for $i = 1,2, \ldots j$, where job (j) is the first tardy job. Let $t(1,k)$ be the maximum among first j jobs. Then with job (k) as shift job, the accomplishment of objectives (1) and (2) stated in Section 2.6.2 is checked. If both are accomplished the shifting is performed. Otherwise, job (j) becomes the shift job and the shifting operation is performed. This rule of selecting the shift job forms the basis of the present heuristic. The various steps of this algorithm are identical to heuristic - 1 except step 3. Step 3 of the heuristic algorithm 3 is given below :

Step-3 : Identify the job having the maximum value of $t(1,i)$ for $i = 1,2, \ldots j$. Let this job be (k). Go to step-4.

2.6.5 Heuristic - 4 :

The information of the idle time for the last machine, i.e., time span during which the last machine is idle and waiting for a job to be processed is very important. The idle time on the last machine for any job reflects the effect of its processing times on all the m machines on its completion time. If a job has a larger processing time on anyone of the m machine it would result in a larger idle time on the last machine. Thus, the vital information of idle time on the last machine. The , point to be borne in mind while exploiting this important and

useful information is that the idle time on the last machine does not reflect the effect of the processing time on the last machine. For example, there could be a job in the sequence for which the processing times on all the machines but for the last machine are smaller in magnitude and as a result it produces very little idle time on the last machine. But, if, this job has a larger processing time on the last machine, the completion time of the job will be large and this will never be reflected by the idle time on the last machine. Therefore, sum of the idle time and the processing time on the last machine of the various jobs is taken as the criterion to identify the shift job. In this heuristic, the job having the maximum value of the sum of idle time and the processing time on the last machine is selected as the shift job. The sum of the idle time and processing time on the last machine could be viewed as the modified processing time on the last machine, meaning that job (k) virtually requires time equal to the sum of the above two quantities on the last machine. Either the last machine is waiting for the arrival of job (k) or processing it. If the above mentioned sum for job (j) is denoted by IT (j), then the expression to calculate it can be derived as follows : Following figure gives a Gantt chart for a two machine, four job problem.

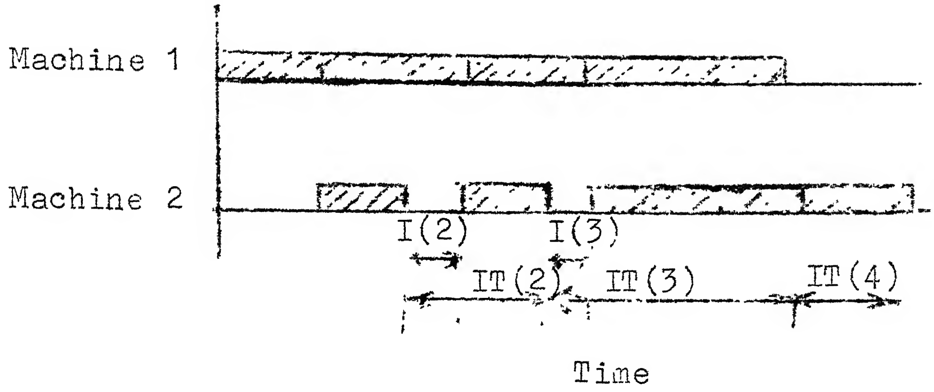


Fig. 2 Gantt chart for a 2-machine 4 job problem

From the figure 2 given above, it can be seen that

$$IT(2) = C(2,2) - C(2,1) = t(2,2) + I(2)$$

$$IT(3) = C(2,3) - C(2,2) = t(2,3) + I(3)$$

$$IT(4) = C(2,4) - C(2,3) = t(2,4).$$

From above we can write a general expression for IT(2) as follows :

$$IT(k) = C(m,k) - C(m,j-1) \text{ for } k = 2,3, \ldots n.$$

It should be noted that IT(1) is not defined because C(m,0) does not exist. Therefore, if the first late job occurs for $j = 2$, then the accomplishment of objectives (1) and (2) is checked with job (1) as the shift job. If both the objectives are accomplished, the shifting operation is performed. Otherwise, job (2) becomes the shift job and the shifting operation is carried out. If the first late job does not occur for $j = 2$, then the job having the maximum value of IT(k) for $k = 2,3, \ldots j$ is selected as the shift job and then a check is made to see if

the objectives (1) and (2) are achieved. If both the objective 8 are achieved job (k) stays as the shift job, otherwise job (j) is selected as the shift job in place of job (k) and the shifting operation is carried out. The procedure mentioned above of selecting the shift job forms the basis of the present heuristic. The various steps of this algorithm are identical to heuristic-1 except step 3. Step 3 of the heuristic algorithm-4 is given below :

Step-3 : If j = 2, Let k = 1; go to step-4; otherwise if j > 2, then identify the job having the maximum value of IT (k), for k = 2,3, ... j. Let this be job (k). Go to step-4.

The common flow chart for all the heuristic algorithms is given in Appendix A.

2.8 ILLUSTRATED EXAMPLE

The suggested heuristics are illustrated with the help of a 2 machine, 5 job problem. The data for the problem is given in Table-I. Step-wise functioning of the various algorithm is given below :

2.8.1 Solution by Heuristic-1

The step-wise functioning of the algorithm is given in Table - II.

Table II : Step-wise progress of Heuristic-1

STAGE-1

Step-1	Initialize. E = {1-2-3-5-4} ; L = ∅
Step-2	Job 3 is the first tardy job in E
Step-3	S(3) has the maximum value among the jobs in E
Step-4	Job 3 is removed from E. E = { 1-2-5-4 } ; L = { 3 }
Step-5	Completion times of all the jobs in E are revised

STAGE-2

Step-2	Job 4 is the first tardy job
Step-3	S(4) has the maximum value among the jobs in E
Step-4	Job 4 is removed from E. E = { 1-2-5 } ; L = { 3,4 }
Step-5	Completion times of all the jobs in E are revised

STAGE-3

Step-2	No jobs in E are tardy. Solution is, { 1-2-5-3-4 } or { 1-2-5-4-3 } and $N_T = 2$.

In Stage-1, the jobs are arranged by EDD order and job 3 is found to be the first tardy job. The shift job upto and including job 3 is job 3. Therefore, it is shifted to L. Similarly, at Stage-2 job 4 is shifted to L. At Stage-3 no tardy jobs remain in E. Therefore, the algorithm yields two alternate sequences, viz.,

1-2-5-3-4 and 1-2-5-4-3, as the jobs in L can

appear in any order. For both the above sequences, $N_T = 2$.

2.7.2 Solution by Heuristic-2 :

The various steps involved in the application of this heuristic are the same as that of heuristic-1 except Step-3. Applying Step-3 of this heuristic it is found that the same jobs as selected by heuristic-1 are chosen as shift jobs viz., job 3 and 4 in stage 1 and 2 respectively. In Stage-3 N tardy jobs remain in E. Therefore the algorithm yields the same alternate sequences as that of heuristic-1. The step-wise progress of the algorithm is same as that of heuristic-1 which is given in Table II.

2.7.3 Solution by Heuristic-3 :

The various steps involved in the application of this heuristic are the same as that heuristic-1 except step-3. Applying step-3 of this algorithm it is found that job 2 has the maximum processing time on the first machine. It is removed from E and N_{T1}, N_{T2}, $C(m,j-1)_1$ and $C(m_1,j-1)_2$ are calculated. As $C(m\ ,j-1)_2 > C(m,j-1)_1$ job 3 becomes the shift job and is shifted to set L. Stage-2 and Stage-3 are identical to that of heuristic-1.

The solution obtained by this heuristic is the same as that obtained by the other heuristics. Step-wise progress of this algorithm is given in Table II-A.

TABLE - I.

Processing times and due date data for the illustrated example

	Job k				
	1	2	3	4	5
t_{1k}	1	2	4	9	7
t_{2k}	3	9	4	7	8
d_k	5	13	14	20	22

TABLE II-A

Step-wise Progress of Heuristic-3

STAGE-1

Step-1 and 2 are the same as that of Table II

Step-3: t(1,2) has the maximum value among the job in E

Step-6: N_{T2} and $C(m,j-1)_2$ are calculated by removing job 2 from E
$N_{T1} = N_{T2}$ and $C(m, j-1)_1 < C(m, j-1)_2$. Therefore job 3 is shifted to set L. E = {1-2-5-4} and L = {3}.

Step-5: Completion times of all the jobs in E are revised.

Stage-2 and 3 are identical to that of Table II.

The solution is 1-2-5-4-3 or 1-2-5-3-4.

2.7.4 Solution by Heuristic-4

The various steps involved in the application of this heuristic are the same as that of heuristic-1. By applying step-3 of this algorithm it is found that, by chance, this heuristic also selects the same jobs as the shift jobs as selected by heuristic-1 at each stage.

Job 3 and job 4 get selected as the shift jobs in stage-1 and 2 respectively. At stage-3 no tardy jobs remain in E. Therefore the algorithm yields the same solution as that of

heuristic-1. The step-wise progress of the algorithm is same as that of heuristic-1 which is given in Table II.

2.8 APPLICATION OF VARIOUS HEURISTICS TO THE ORDERED FLOW SHOP PROBLEM

Another class of scheduling problems which has attracted the attention of researchers recently is the mxn flow shop problem with ordered processing time matrices. This problem has two unique characteristics.

i) If a particular job has a smaller processing time on any machine than any machine than any other job, then the processing time of the former job will be less than or equal to the processing time of the later job on all corresponding machines.

ii) The machine with the minimum processing time for a given job will also have the minimum processing time for every other job.

Smith et al [19] have suggested a methodology which gives the optimal solution considering the make-span as the effectiveness criterion for a special class of the above problem for which the maximum processing times occur either on the first or the last machine.

In the present work, the heuristics developed in Section 2.6 are applied to a special class of the ordered flow shop problem for which the following additional constraints are valid.

1) Processing times of any job k are in non-decreasing order of magnitude i.e., $t(1,k) \leq t(2,k) \leq \quad \leq t(m,k)$

2) $\min\limits_{1 \leq k \leq n} \{ t(j,k) \} \geq \max\limits_{1 \leq k \leq n} \{ t(i,k) \}$ for $i < j$

As the processing times are in nondecreasing order, any machine will not have idle time. Therefore, if (j) is the first tardy job, then,

$$C(i,j) = C(i,j-1) + t(m,j) > d_j \tag{7}$$

$$C(i,j-1) \leq d(j-1)$$

$$\text{but } C(i,j-1) = \sum_{i=1}^{m-1} A(i,1) + \sum_{k=1}^{j-1} t(m,k) \tag{8}$$

$$\sum_{i=1}^{m} t(i,1) + \sum_{k=1}^{j-1} t(m,k) + t(m,j) > d(j) \tag{9}$$

Substituting for C(i,j-1) in eqn. (7) and (8), we get,

$$\sum_{i=1}^{m} t(i,1) + \sum_{k=1}^{j-1} t(m,k) \leq d(j-1) \tag{10}$$

The job having the maximum value of t(m,k) for k = 1,2, ... j, is chosen as the shift job. It is assured that after the removal of the shift job from E atleast job (j) would become nontardy as its completion time would be reduced by $\max\limits_{1 \leq k \leq j} \{ t(m,k)\}$: sotherwise if the maximum occurs for i = j, then job (j) would be shifted to L. Because of the special structure of the problem all the four heuristics would choose the same shift job and hence proceed identically. Further the special structure of the problem results in absence of idle times and thus all the four heuristics will yield optimum solutions.

CHAPTER 3

RESULTS AND DISCUSSIONS

A composite computer code was developed for all the four heuristics described in Chapter 2. The programme was written in FORTRAN-10 and implemented on DEC-1090 time sharing computer system. The computer listing of the programme is given in Appendix B.

The performances of the various heuristics were evaluated for 60 randomly generated problems of 12 different sizes. The number of machines considered were 2,3,4 and 5 while the number of jobs considered were 10,25 and 40. For each problem size 5 randomly generated problems were considered.

3.1 GENERATION OF RANDOM PROBLEMS

For each job, its due dates and the processing times on various machines were generated randomly from a uniform distribution. The uniform distribution was assigned an upper limit on the processing times of jobs on various machines for each problem size. Similarly, the uniform distribution for determining the due dates was given an upper limit. The selected upper limits are given in Table 3. The lower limits for both due date and processing times were kept at zero.

3.2 RESULTS

3.2.1 Performance of Heuristics for General Flow Shop Problem

Table 4 gives the relative performance of the various heuristics. It is observed that heuristics-1,2,3 and 4 yielded the best solutions with a frequency of 7,14,17 and 17, respectively. This indicated that heuristic-3 and 4 are marginally superior to heuristic-2. Further, the performance of the heuristic-1 to generate the best sequences was inferior amongst all the four heuristics considered. Of the 60 problems considered, all the four heuristics resulted in same sequences for 15 problems.

Table 5 gives the relative performance of the various heuristics and EDD considering number of problems for which the optimality of the sequence is guaranteed (N_T = 0 or 1). Heuristics-1 and 2 resulted in guaranteed optimal solution for three additional problems while for heuristics-3 and 4 the number was 4 additional problems.

The relative performance of the various heuristics considering the average number of tardy jobs (average based on the five problems of the same size) is given in Table-6. At the bottom of the table for each heuristic, the overall average number of tardy jobs considering all the 12 problem sizes is also given. It is observed that based on the criterion the performance of the heuristic 4 is the best followed by heuristic-2,3 and 1.

The larger size problem took about 102 milliseconds of execution time.

It was observed that the computational time increases significantly with an increase in the number of jobs keeping the number of machines constant. However, the increase was not significant as the number of machines were increased keeping the number of jobs constant.

3.2.2 Performance of Heuristics for Ordered Flow Shop Problem

The performance of all the four heuristics was tested on 3 ordered flow shop problems given in Tables 10-12. These problems carry the special structure, namely: 1) the processing time of all the jobs on machine 1 is less than the processing time of the same job on 2nd machine in the sequence; 2) the minimum processing time on the 2nd machine is greater than or equal to the maximum processing time on machine 1.

For each problem, all the four heuristics yielded the same results. The number of tardy jobs for problems 1,2 and 3 were 1,4 and 4, respectively. The application of EDD rule gave 2,4 and 6 as the number of tardy jobs for problems 1,2 and 3, respectively.

3.3 CONCLUSIONS

In this thesis, four heuristics are developed for the general mxn flow shop scheduling problem for the minimimization of the total number of tardy jobs. The performances of these heuristics were checked for randomly generated problems of 12 different sizes. For each problem size, 5 problems were considered. The performances of the heuristics were evaluated based on average number of tardy jobs, number of times best solution was generated, the frequency of obtaining guaranteed optimal solution and the average execution time for various problem sizes. The analysis indicates that, heuristic-4 gives overall best performance.

As theoretically envisaged, all the four heuristics yielded same results for the 3 specially structured ordered processing times problems.

Further, it was observed that the number of jobs have more significant effect on the computational times than the number of machines. In case of all the heuristics the average execution time is very small. It is suggested that a problem should be solved using all the four heuristics and the best solution should be adapted since no heuristic explicitly guarantees the best solution.

Table 3

Upper Limits on Processing Times and Due Dates
for Various Problem Sizes Considered

Problem size	Upper limit on processing times	Upper limit on due dates
2x10	30	250
2x25	10	250
2x40	20	700
3x10	30	450
3x25	10	300
3x40	20	900
4x10	30	750
4x25	10	450
4x40	20	1600
5x10	30	750
5x25	10	600
5x40	20	1300

Table 4

Relative Performance of Heuristics

Problem size mxn	No. of problems conside-red	No. of problems for which the best solution was obtained by heuristics No. 1	2	3	4	No. of problems for which all the heuristics gave the same solution
2x10	5	2	0	0	0	2
2x25	5	0	1	1	3	2
2x40	5	1	3	3	3	1
3x10	5	1	1	0	1	1
3x25	5	0	1	1	2	3
3x40	5	1	0	2	0	2
4x10	5	0	1	1	1	1
4x25	5	1	0	3	1	1
4x40	5	0	2	1	1	1
5x10	5	0	1	0	1	1
5x25	5	0	2	2	3	0
5x40	5	1	2	3	1	0

Table 5

Relative performance of heuristics of EDD considering number of problems for which optimably of the sequence is guaranteed (i.e. N_T = 0 or 1)

Problem size mxn	Heuristic No. 1	2	3	4	EDD
2x10	1	1	1	1	1
2x25	0	0	0	0	0
2x40	0	0	0	0	0
3x10	3	2	2	2	1
3x25	0	0	0	0	0
3x40	0	0	0	0	0
4x10	3	4	4	4	2
4x25	0	0	0	0	0
4x40	0	0	0	0	0
5x10	3	3	3	3	3
5x25	0	0	1	1	0
5x40	0	0	0	0	0
Total	10	10	11	11	7

Table 6

Relative performance of various heuristics considering average number of tardy jobs

Problem size mxn	Average No. of tardy jobs Heuristic No. 1	2	3	4
2x10	1.8	2.2	2.4	2.2
2x25	5.8	5.6	5.4	5.0
2x40	12.4	11.8	11.8	11.8
3x10	1.8	1.6	2.0	1.6
3x25	5.0	4.8	4.8	4.6
3x40	8.8	9.4	8.4	9.6
4x10	1.2	1.0	1.0	1.0
4x25	4.4	4.4	4.4	3.8
4x40	5.8	5.0	5.4	5.6
5x10	1.0	0.8	1.0	0.8
5x25	3.8	3.2	3.2	2.8
5x40	5.8	5.6	5.4	5.8
Overall Average No. of tardy jobs	5.8	4.6	4.6	4.5

Table 9

Average execution times for problems of different sizes

Problem size	No. of problems	Average time taken by Heuristic (in milliseconds) 1	2	3	4
2x10	5	10	*	10	*
2x25	5	21	18	39	19
2x40	5	74	90	102	72
3x10	5	*	*	14	11
3x25	5	32	31	34	35
3x40	5	32	84	93	90
4x10	5	*	14	*	14
4x25	5	41	36	49	58
4x40	5	78	70	72	71
5x10	5	12	*	13	13
5x25	5	40	56	60	84
5x40	5	83	82	121	98

* denotes the average execution time less than 10 milliseconds.

Table 10

Data for the ordered flow shop problem No.1

	Job k 1	2	3	4	5	6
t_{1k}	5	10	12	14	15	18
t_{2k}	26	30	35	40	50	55
d_k	150	140	150	200	150	300

Table 11

Data for the ordered flow shop problem No. 2

	Job k 1	2	3	4	5	6	7	8	9	10
t_{1k}	33	32	32	31	30	25	13	12	11	10
t_{2k}	85	75	72	70	65	60	42	40	38	33
d_k	150	280	390	500	250	200	220	160	140	320

Table 12

Data for the ordered flow shop problem No. 3

	Job k						
	1	2	3	4	5	6	7
t_{1k}	40	42	44	46	48	49	50
t_{2k}	50	60	70	80	90	95	100
d_k	160	350	300	250	180	150	200

REFERENCES

1. Bakshi, M.S. and S.R. Arora, 'The Machine Sequenching Problem', Management Science, Vol. 16, No.4, 1969, pp. B 247-263.

2. Johnson, S.M., 'Optimal Two and Three Stage Production Schedules with Set-up Times Included', Naval Research Logistic Quarterly, Vol. 1, No.1, 1954, pp. 61-68.

3. Cambell, H.G., Richard A. Dudek and Milton L. Smith, 'A Heuristic Algorithm for the n-job m-machine sequenching Problem', Management Science, Vol. 16, No.2, 1970, pp. B 630-637.

4. Palmer, D.S., 'Sequencing Jobs Through a Multi-Stage Process in the Minimum Total Time - A Quick Method of Obtaining a Near Optimum', Operational Research Quarterly, Vol. 23, No.3, 1972, pp. 323-331.

5. Gupta, J.N.D., 'A Functional Heuristic Algorithm for the Flow-shop Scheduling Problems', Operational Research Quarterly, Vol. 22, No.1, 1971, pp. 39-48.

6. Petrov, V.A., Text Book, Flow Line Group Production Planning, London, Business Publications, 1968.

7. Nobeshima, I., 'The Order of n items Processed on m-Machines', Journal of Operations Research Society of Japan, Vol. 16, No.3, 1973, pp. 163-185.

8. Krone, M.J. and Kenneth Steiglitz, 'Heuristic Programming Solution of a Flow-Shop Scheduling Problem', Operations Research, Vol. 22, No.3, 1974, pp. 621-628.

9. Page, E.S., 'An Approach to Scheduling Jobs on Machines', Journal of the Royal Statistical Society, Series B, Vol. 23, pp. 484-492.

10. Maxwell, W.L., 'On Sequencing n jobs on One Machine to Minimize the Number of Late Jobs', Management Science, Vol. 16, No.5 (January, 1970).

11. Sydney, J.B., 'A Comment on a Paper of Maxwell', Management Science, Vol. 18, No.11, pp. 715-717.

12. Maxwell, W.L., 'A Rejoinder', Management Science, Vol. 18, No.11, pp. 718-

13. Moore, J.M., 'A Comment on a Paper of Maxwell by J.B. Sydney - A Rejoinder', Management Science, Vol. 18, No.11, pp. 720.

14. Moore, J.M., 'Sequencing n Jobs on One Machine to Minimize the Number of Late Jobs', Management Science, Vol. 15, No.1 (September, 1968).

15. Strum, L.J.M, 'A Simple Optimality Proof of Moore's Sequencing Algorithm', Management Science, Vol. 17, No.1, (September, 1970).

16. Elmagraby, S.E., 'On the Sequencing of n-Jobs on one Machine to Minimize the Number of Jobs Late - A Rejoinded Management Science, Vol. 18, No.11, pp. 389.

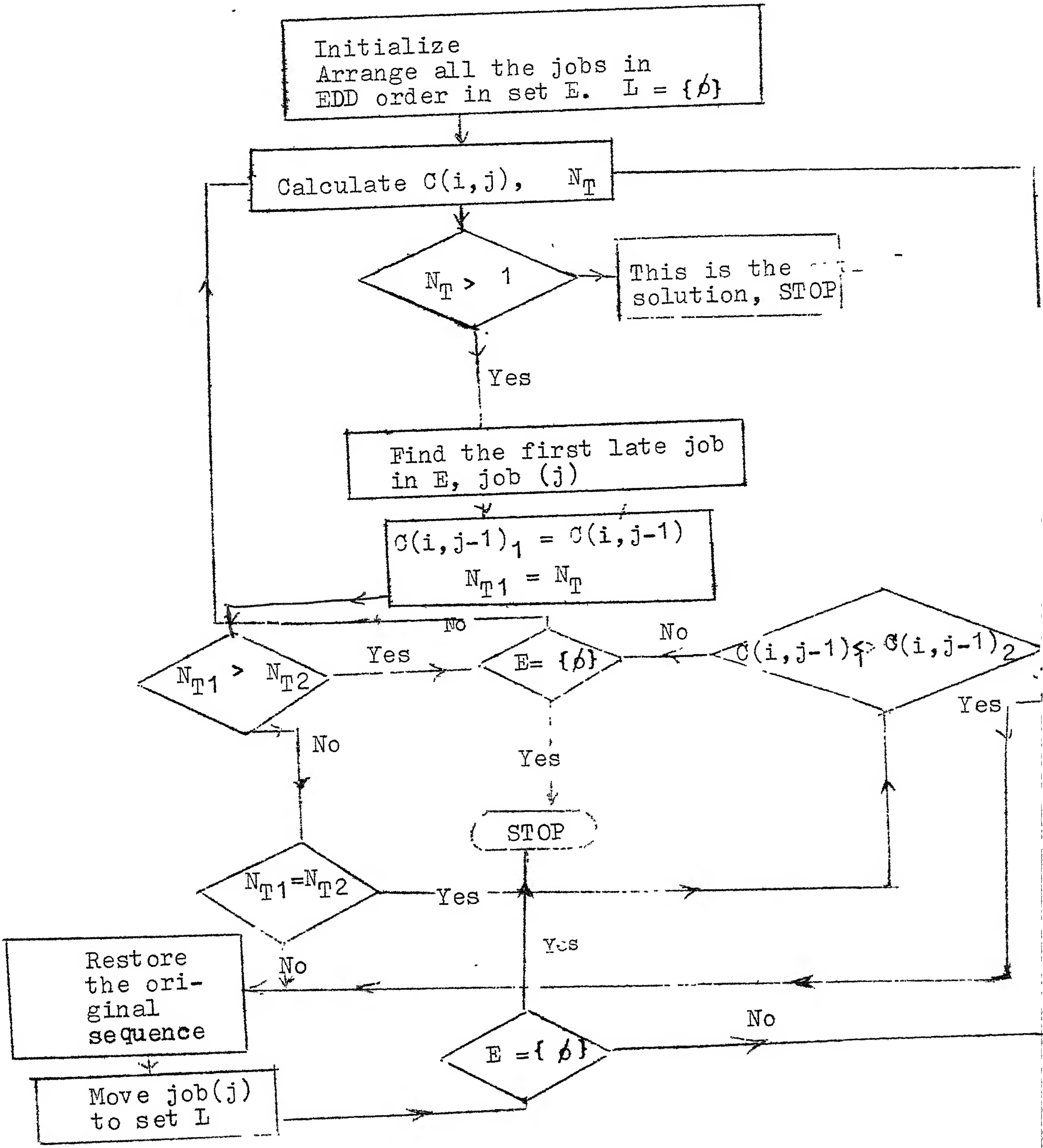
17. Baker, K.R., 'Introduction to Scheduling and Sequenching', John Wiley & Sons, Inc., 1974, pp. 305.

18. Baker, K.R. and Martin, J.B., 'An Experimental Comparison of Solutions Algorithms for the Single - Machine Tardiness Problem', Naval Research Logistic Quarterly, Vol. 21, No.1, (March, 1974).

19. Smith, M.L., Dudek, R.A. and Panwalkar, 'Flow Shop Sequencing Problem with Ordered (Scheduling) Processing Time Matrices', Management Science, Vol. 21, No.5, pp. 544.

APPENDIX A

FLOW CHART FOR THE PROPOSED HEURISTICS



```
C
C
C
C
C
C     PROGRAM TO MINIMISE NO. OF
C           TARDY JOBS FOR  M-M/C N-JOB FLOW SHOP.
            IMPLICIT INTEGER(A-Z)
            DIMENSION EDD(50),TAV(5),B(5),NT(5)
            COMMON NT,TARNES,A,N,LL,T(0:10,0:50)
            1,PP(50),C(0:6,0:50
            2),TOTALT(50),D(50),IOUT,IT(50),LLTAR(0:50)
            DATA INPUT,IOUT,IOUT1/20,21,22/
            OPEN(UNIT=INPUT,FILE='MN.DAT')
            OPEN(UNIT=IOUT1,FILE='TT.OUT',ACCESS='APPEND')
            OPEN(UNIT=IOUT,FILE='IT2.OUT',ACCESS='APPEND')
C     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
C           FOR THE EXPLATION OF THE VARIABLE NAMES USED PL.
C           REFER  SECTION 2.2.
C     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
            READ(INPUT,*)NOOFPR
            DO 333 FFF=1,NOOFPR
            READ(INPUT,*)M,N
            DO 10 J=1,N
            READ(INPUT,*)(T(I,J),I=1,M),D(J)
            PP(J)=J
10          CONTINUE
C           SUBROUTINE VSRTPM ARRANGES THE
C               JOBS IN EDD ORDER.
            CALL VSRTPM(D,N,PP)
            DO 28 R=1,N
28          EDD(R)=PP(R)
C           WRITE(IOUT,313)(PP(J),J=1,N)
C           WRITE(IOUT,323)((I,T(I,J),J=1,N),I=1,M)
C323        FORMAT(10X,'M/C-NO.',15X,'JOB-NO.'/20X,50(I2,5X))
C313        FORMAT(11X,50(I3,4X))
            LTAR=100
            CALL COMPLN
            CALL TARDY
            WRITE (IOUT,15)NT
15          FORMAT(50X,'EDD-NT = ',I5)
            LLTAR(0)=100
            LTAR=NT
            [illegible]
C     ***********************************
C        GIVEN THE JOBS IN EDD SEQ.
C         THIS [illegible] FINDS THE OPT. SEQ.
C        BY APPLYING [illegible] ALGORITHM.
C     ***********************************
            DO [illegible]=1,4
            CALL [illegible]
C           IF([illegible].GT.1)[illegible]334
            DO [illegible] S=1,[illegible]
```

```
C
C
C
C
C
C
C     PROGRAM TO MINIMISE NO. OF
            TARDY JOBS FOR N-M/C M-JOB FLOW SHOP.
            IMPLICIT INTEGER(A-Z)
            DIMENSION EDD(50),TAV(5),B(5),NT(5)
            C[illegible]T,TARDES,[illegible],LL,T(0:10,0:50)
            1,PP(50),C(0:6,0:50
            2),[illegible]TALT(50),D(50),IOUT,IT(50),LLTAR(0:50)
            DATA INPUT,IOUT,IOUT1/20,21,22/
            OPEN(UNIT=INPUT,FILE='[illegible].DAT')
            OPEN(UNIT=IOUT1,FILE='TT.OUT',ACCFSS='APPEND')
            OPEN(UNIT=IOUT,FILE='IT2.OUT',ACCESS='APPEND')
C           ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
C           FOR THE EXPLATION OF THE VARIABLE NAMES USED PL.
C           REFER  SECTION 2.2.
C           ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
            READ(INPUT,*)NOEPR
            DO 333 EEE=1,NOEPR
            READ(INPUT,*)M,N
            DO 10 J=1,N
            READ(INPUT,*)(T(I,J),I=1,M),D(J)
            PP(J)=J
10          CONTINUE
C           SUBROUTINE VSRTPM ARRANGES THE
C              JOBS IN EDD ORDER.
            CALL VSR[illegible](D,N,PP)
            DO 28 R=1,N
28          EDD(R)=PP(R)
C           WRITE(IOUT,313)(PP(J),J=1,N)
C           WRITE(IOUT,323)((I,T(I,J),J=1,N),I=1,M)
C323        FORM[illegible]T(10X,'[illegible]',15X,'JOB-NO.'/20X,50(I2,5X))
C313        F[illegible](I1[illegible](I3,4X))
            [illegible]
            CALL [illegible]
            [illegible]
            WRITE (IOUT,15) [illegible]
15          FORMAT(5[illegible]X,'[illegible] = ',I5)
            [illegible]
            [illegible]
            [illegible]
C     ***********************************
C        GIVE THE JOBS IN EDD SEQ.
C         THIS PORTION FINDS THE OPT. SEQ.
C        BY APPLYING [illegible]'S ALGORITHM.
C     ***********************************
            DO [illegible] CODE=1,4
            CALL [illegible]
C           IF((CODE.GT.1)[illegible]
            DO 23 S=1,N
```

```
23        PP(S)=EDD(S)
          LTAR=FDDNT
          CALL COMPLN
          NN=N+1
20        NN=NN-1
          IF (NN.LE.1)300,400
400       CALL COMPLN
 3070     DO 40 I=1,NN
          IF (((C(M,PP(I))-IT(PP(I))).GT.0)30,40
30        MAX=0
          LL=I
          IF (I.EQ.1)  145,420
145       LK=1
          GO TO 7070
 420      IF(I.EQ.2)34,460
34        LK=1
          GO TO 165
460       GO TO(4800,4700,4600,4900)CODE
4700      CALL TTMC2
          GO TO 160
4800      CALL TTMC1
          GO TO 160
4900      CALL IT2
          GO TO 160
4600      CALL TOTAL
160       DO 140 IJK=2,I

          IF (IT(PP(IJK)).GT.MAX)130,140
C
C
C
C
C
C
130       MAX=IT(PP(IJK))
          LK=IJK
140       CONTINUE
165       PP(N+1)=PP(LK)
          LCOMP=C(M,PP(N+1))
          DO 170 A=LK,N
170       PP(A)=PP(A+1)
          CALL COMPLN
          CALL TARDY
          IF (NT-LTAR)625,625,1010
625       IF(LCOMP.GE.C(M,PP(LK))) GO TO 1020
 1010     DO 1030 BBB=N,LK+1,-1
 1030     PP(BBB)=PP(BBB-1)
          PP(LK)=PP( +1)
          LK=I
 7070     PP(N+1)=PP(LK)
          DO 1700 A=LK,N
1700      PP(A)=PP(A+1)
          CALL COMPLN
          CALL TARDY
 1020     LTAR=NT
C         WRITE(5,1234)(PP(J),J=1,N)
C1234     FORMAT(10X,50(I2,'-*'))
          GO TO 20
40        CONTINUE
300       CALL COMPLN
```

Page:

```
C           WRITE(IOUT,789)((C(M,JJ),JJ=1,N)
C789        FORMAT(10X,50(I3,'-'))
C12         FORMAT(10X,'COMPLN.-TIME.**!!!**=',I5)
            CALL TARDY
            LGTAR(KKK)=NT
C           WRITE (IOUT,25)KKK,(PP(I),I=1,N)
C25         FORMAT(10X,'JOB-SEQ. AT THE END OF
C           1ITTERATION NO.-',I2,' IS AS FO
C           1LLOWS'/10X,50(I4)//)
            WRITE (IOUT,55)NT
 55         FORMAT(10X,' NT = ',I4)
            IF (NT.GE.1)510,550
550         IF (LGTAR(KKK)-LGTAR(KKK-1))600,700,800
800         WRITE(IOUT,150)NT
150         FORMAT(10X,'**ERROR . NO. OF TARDY
            1JOBS GETS INCREASED.& IT IS
            1=',I4)
            GO TO 111
C700        CONTINUE
700         GO TO 111
600         LTAR=NT
510         CONTINUE
111         CALL RTIME(TIME2)
C           WRITE(IOUT,520)(PP(I),I=1,N)
C520        FORMAT(15X,'OPT.  SEQ. IS AS FOLLOWS
            1C'/18X,50(I4)//)
            TIME3=(TIME2-TIME1)
            WRITE(IOUT1,88)CODE,TIME3
88          FORMAT(10X,I1,I6)
222         FORMAT(10X,'END OF PRO. NO.--',I5)
89          CONTINUE
            WRITE(IOUT,222)FFF
            WRITE(IOUT1,222)FFF
333         CONTINUE
500         WRITE(IOUT,595)COFPN
            WRITE(IOUT1,595)COFPN
595         FORMAT(30X,'END OF ',I5,'  PRO. SESSION')
39          CONTINUE
            STOP
            END
```

```
C
C
C
C
C
C
          SUBROUTINE COMPL
          IMPLICIT INTEGER(A-Z)
          COMMON TARDYN,TARNES,M,N,LL,T(-5:10,0:50),
          1PP(50),COMP(0:6,0:50
          2),TOTALT(50),D(50),IOUT
          DO 5 KK=1,M
          COMP(KK,PP(1))=COMP(KK-1,PP(1))+T(KK,PP(1))
5         CONTINUE
          DO 10 K=1,M
          DO 10 J=2,N
          IF(COMP(K,PP(J-1)).GT. COMP(K-1,PP(J)))300,400
300       COMP(K,PP(J))=COMP(K,PP(J-1))+T(K,PP(J))
          GO TO 10
400       COMP(K,PP(J))=COMP(K-1,PP(J))+T(K,PP(J))
10        CONTINUE
C         WRITE(IOUT,12)COMP(M,PP(N))
C12       FORMAT(10X,'COMPL.-TIME.~~!!!~~=',I5)
          RETURN
          END
~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~
          SUBROUTINE TARDY
          IMPLICIT INTEGER(A-Z)
          COMMON TARDYN,TARNES,M,N,LL,T(-5:10,0:50),
          1PP(50),COMP(0:6,0:50
          1),TOTALT(50),D(50),IOUT
          TARDYN=0
          TARNES=0
          DO 10 I=1,N
          IF ((COMP(M,PP(I))-D(PP(I))).LE.0)10,20
20        TARDYN=TARDYN+1
          TARNES=TARNES+(COMP(M,PP(I))-D(PP(I)))
10        CONTINUE
C         WRITE(IOUT,93)TARDYN
C93       FORMAT(10X,'~~~~  NO. OF TARDY JOBS'/16X,I4)
C         WRITE(IOUT,50)TARNES
C50       FORMAT(10X,'~~TOTAL TARDYNESS FOR ABOVE SEQ. IS =',I4/)
C         WRITE(IOUT,555) COMP(M,PP(N))
C  555    FORMAT(15X,'~~~~ COMPLETION TIME =',I3)
          RETURN
          END
~~~~~~~~~~~~~~~~~~
C~~~~~~~~~~~~~~~~~~
          SUBROUTINE ITS
          IMPLICIT INTEGER(A-Z)
          COMMON TARDYN,TARNES,M,N,LL,T(-5:10,0:50),
          1PP(50),COMP(0:6,0:50
          1),TOTALT(50),D(50),IOUT
```

```
            DO 10 J=1,2
            TOTALT(PP(J))=0
            DO 10 K=1,N
10          TOTALT(PP(J))=TOTALT(PP(J))+I(K,PP(J))
C           WRITE(IOUT,25)(TOTALT(PP(J)),J=1,N)
C25         FORMAT(40X,'TOTAL-TIME'/10X,50(I4,'--'))
            RETURN
            END
C        ~~~~~~~~~~~~~~~~~~~~~~
```

```
C~~~~~~~~~~~~~~~~~~
C~~~~~~~~~~~~~~~~~~
        SUBROUTINE TTMC1
        IMPLICTT INTEGER(A-Z)
        COMMON TARDYN,TARNES,M,N,LL,T(-5:10,0:50),
        1PP(50),CUMP(0:6,0:50
        1),TOTALT(50),D(50),IOUT,IT(50)
        DO 10 J=1,N
10      IT(PP(J))=T(1,PP(J))
        RETURN
        END
C     ~~~~~~~~~~~~~~~~~~~
 C    ~~~~~~~~~~~~~~~~~~~
        SUBROUTINE TTMC2
        IMPLICTT INTEGER(A-Z)
        COMMON TARDYN,TARNES,M,N,LL,T(-5:10,0:50),
        1PP(50),COMP(0:6,0:50
        1),TOTALT(50),D(50),IOUT,IT(50)
        DO 10 J=1,N
10      IT(PP(J))=T(M,PP(J))
        RETURN
        END
C     ~~~~~~~~~~~~~~~~~~~
 C    ~~~~~~~~~~~~~~~~~~~
C~~~~~~~~~~~~~~~~~~
C~~~~~~~~~~~~~~~~~~
        SUBROUTINE IT2
        IMPLICIT INTEGER(A-Z)
        COMMON TARDYN,TARNES,M,N,LL,T(-5:10,0:50),
        1PP(50),COMP(0:6,0:50
        1),TOTALT(50),D(50),IOUT,IT(50)
        DO 10 J=1,N
10      IT(PP(J))=C(M,PP(J))-C(M,PP(J-1))
        RETURN
        END
C     ~~~~~~~~~~~~~~~~~~~
 C    ~~~~~~~~~~~~~~~~~~~
```

```
C        PROG. TO GENERATE RANDOM DATA FOR,M/N M-C CASE.
         IMPLICIT INTEGER (A-Z)
         REAL R(5000)
         DIMENSION TOTAL(1000),T(5,1000),D(1000)
         DATA IOUT,INPUT,INCR,IOUT1/21,20,0,22/
         OPEN (UNIT=INPUT,FILE='R1.DAT')
         OPEN (UNIT=IOUT,FILE='M1.DAT')
C        OPEN (UNIT=IOUT1,FILE='TIME.OUT',ACCESS='APPEND')
         CALL [illegible](I1SEC1)
         READ(INPUT,*)NSESON
         DO 9999 XYZ=1,NSESON
         READ (INPUT,*)NOOFPR,MSIZE1,MSIZE2,ISIZE1,ISIZE2
        1,ISTEP,LIMDUE,LIMTIM
         ITOTPR=(1+(ISIZE2-ISIZE1)/ISTEP)*
        1(1+(MSIZE2-MSIZE1))*NOOFPR
         WRITE(IOUT,*) NSESON
         WRITE(IOUT,*)ITOTPR
         DO 200 M=MSIZE1,MSTZE2
         DO 150 N=ISIZE1,ISIZE2,ISTEP
         CALL RANDOM(R,RINDEX,INCR)
         DO 100 JJJ=1,NOOFPR
         WRITE(IOUT,201)M,N
201      FORMAT(10X,I4,5X,I4)
         DO 20 J=1,M
         DO 20 I=1,N
111      RINDEX=RINDEX+1
         IF(RINDEX==5000) CALL RANDOM(R,RINDEX,INCR)
         IR=R(RINDEX)*100
         IF (IR.EQ.0) GO TO 111
         IF(IR.GT.LIMTIM) GO TO 111
         T(J,I)=IR
20       CONTINUE
         DO 30  I=1,N
         TOTAL(I)=0
         DO 22 J=1,M
22       TOTAL(I)=TOTAL(I)+T(J,I)
25       RINDEX=RINDEX+1
         IF(RINDEX==5000) CALL RANDOM(R,RINDEX,INCR)
         IR=R(RINDEX)*5000
         IF (IR==0.OR.IR.GT.LIMDUE.OR.IR.LT.TOTAL(I)) GO TO 25
         D(I)=IR
         WRITE(IOUT,*)(T(J,I),J=1,M),D(I)
30       CONTINUE
100      CONTINUE
150      CONTINUE
200      CONTINUE
C        CALL [illegible](IT [illegible]2)
C        [illegible]=[illegible]
C        WRITE(IOUT1,99)TIME3
9999     CONTINUE
99       F[illegible] X,'TIME-TAKEN  =',I6)
         STOP
         END
C      ~~~~~~~~~~~~~~~~~~~~~
C      ~~~~~~~~~~~~~~~~~~~~~
         SUBROUTINE RANDOM(R,RINDEX,INCR)
         DIMENSION R(5000)
         INTEGER RINDEX,INCR,ISED
```

```
INCR=INCR+1
ISED=ISED+123+INCR
J=5000
CALL GGUB(ISED,J,R)
RINDEX=1
RETURN
END
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~